A task framework for the web interface W2H

Peter Ernst*, Karl-Heinz Glatting and Sándor Suhai

DKFZ—German Cancer Research Center, Department of Molecular Biophysics, Im Neuenheimer Feld 280, 69120 Heidelberg, Germany

Received on May 6, 2002; revised on September 9, 2002; accepted on September 17, 2002

ABSTRACT

Summary: The W3H task framework allows the execution of compound jobs utilizing the description of work and data flows in a heterogeneous bioinformatics environment using meta-data information. By means of these descriptions, the task system can schedule the necessary execution of applications available in the environment, depending on rules specified in the meta-data. By integrating this task framework into the publicly available web interface W2H, similarly based on meta-data, web access and data management are immediately available for each task description. Authors of task descriptions can base their work on the underlying classes and objects to be able to describe dependency rules between previously independent applications. The result of a compound task is given as XML data that is translated according to XSLT into web pages or plain text to report the result of the task to the user.

Availability: Within the HUSAR environment at DKFZ http://genome.dkfz-heidelberg.de/.

Contact: P.Ernst@dkfz.de

INTRODUCTION

In high-throughput sequence analysis it is often necessary to combine the results of contemporary bioinformatics tools, because no individual tool alone computes all of the requested information.

Likewise because of the variety of the available tools and methods, it is often difficult for a user of bioinformatics to select the right tool with the right parameters for the current question. For this user’s assistance a local task designer should create a scheme to automatically select the right tool depending on properties of the input data and additional questions answered by the user prior to the tool selection. Such a task designer needs to have a well-founded background in life sciences to be able to combine the methods implemented in the available tools.

The common solution for this kind of problem is to create a wrapper program that prepares the input for the individual analysis tools. Then the wrapper calls the tools and finally reads their outputs to create a report of the whole analysis.

One of the disadvantages of this procedure is the lack of reusability for situations where a 2nd, 3rd or more tasks need to be created. Projects like BioPerl, BioJava or BioPython (OpenBio, 2002) try to address this problem but they still require the knowledge of a programming language. In this article we describe the development of a task framework that reduces the amount of necessary programming skills for a task author as much as possible and contains a concept of reusability for the written code.

However, the most important aspect of this framework is that it seamlessly integrates into the web interface W2H (Senger et al., 1998). Because of the meta-data concept used in W2H and this task framework, knowledge in web programming is not required for bringing custom tasks onto the web.

SYSTEM AND METHODS

The W3H task framework was developed in the HUSAR (Senger et al., 1995) environment at the DKFZ (German Cancer Research Center)—an environment which includes the tools of the GCG (Devereux et al., 1984) and EMBOSS (Rice et al., 2000) packages and the Phylip suite (Felsenstein, 1995) among others. The W3H task framework was designed to interact with the web interface W2H—a free, popular user front-end for sequence analysis tools.

Development of the web interface W2H started in 1996 as a project to enable web access to the tools of the HUSAR environment. To avoid having to develop a web interface for each individual tool, W2H employs a meta-data approach. This means that providing a description of the parameters of a tool is sufficient to create a web interface for it. The analysis packages from GCG and EMBOSS already include such meta-data. Therefore, W2H can serve not only HUSAR but installations of GCG and EMBOSS as well.

Since the initial version of W2H, several new features (W2H History, 2002) were implemented. These include special interfaces for beginners and advanced users, more options when visualizing graphical data, hyperlinking

*To whom correspondence should be addressed.
of application results with public databases, more data management options and the possibility of starting an analysis on several selected sequences at once (loop feature).

However, W2H was created based on the classical tool-oriented approach (selector of tools). Although it provides a simple interface for high-throughput analysis utilizing a single tool (W2H’s loop feature) the combination of tools is not supported.

In order to integrate applications and methods into a problem-oriented analysis task using a meta-data approach like in W2H, we identified three fields that need to be described:

- **Data flow.** To forward information computed by one application to another, it is necessary to parse the outputs and provide the data that they contain—or pieces thereof—to a subsequent application. Occasionally unaltered results can be used as input for the next application but usually it becomes necessary to extract just pieces of an output and assemble them properly for a following application. To be able to make the work flow dependent on computed results (see below) it is necessary to translate the results from their human readable format into a machine readable format. Only then is it possible to define rules for the task that depend on the outcome of prior computations.

Additionally, if a client-side image map—a figure with hyperlinks—is supposed to be part of the task’s output report, parts of the individual outputs need to be extracted and additional information like web resources need to be added. Then the data can be reassembled to form a description of the map.

- **Work flow and application dependencies.** In order to get reliable results several methods, implemented in different applications, need to be applied. The dependencies between methods and applications follow certain rules that must be described in order to be able to compute the results automatically without human interaction. Using these rules, the task system can decide whether to call some applications in parallel or sequentially if necessary. Moreover, the rules determine the possibility and necessity of calling certain methods depending on previous calculations or user defined settings.

- **Task result.** The final output of a task run should be a report containing the relevant information that was obtained during the runs of the individual applications before. This requires the assessment of the data from the individual results and their combination in a preferably uniform layout, in order to facilitate the user’s comprehension of the information given. Access to some of the individual outputs may be useful in certain situations.

Although the work and data flow could be described in separate meta-data descriptions we found that the overlap between those two parts of a task description was so extensive that describing the work and data flow together can save a lot of work.

Using the existing W2H, the data flow description is used by the web interface to dynamically create the HTML input forms for the task data. With the given meta-data the web interface can collect the input from the user, check if all minimum requirements are fulfilled and provide the data to the task system. W2H additionally uses the data flow description to monitor all data files created during a task run. With this information a data management interface could be created. By means of this interface, it is possible for the user to decide whether to keep or discard the data. Setting up automatic clean-up mechanisms that delete data independently of their importance is therefore not necessary.

Currently we are improving W2H’s understanding of the data flow descriptions in order to allow communication with the user during task runs (interactive tasks).

The work flow description is used by the task system’s **job scheduler** (Figure 1), to initiate the start of the individual applications. The language describing the work flow easily allows the indication of parallel or sequential processing of the tools. The parameter specification for a tool or method follows the tool’s syntax on the UNIX command-line, which is well known to people who want to implement a certain tool and need to create a task description (task administrator). Additionally, there is support for incorporating web-based tools from other sites into a task description. To share the machine load between several computers, the addition of a configurable link between the job scheduler and a 3rd party load sharing software is planned.

The tool’s result(s) will then be processed by the task framework’s **parser engine** which must be furnished with parser plug-ins for each specialized tool output. The parser engine itself provides an object-oriented tool box that supports the creation of parser plug-ins. The outcome of this process is the transformation of the tool’s result into an object, where public methods give access to the object’s properties. In this case it means that the methods provide the computed data from the bioinformatics tool. These methods are used in the work flow descriptions to implement conditional rules defining how to proceed if a certain result occurred or not.

For the final task report and in interactive tasks, the task system stores the objects in XML format on disk. This data is transformed by means of W2H’s post-processing mechanism into an HTML page for the task report using
implement the public GD library (GDlib, 2002) to create maps in HTML from sequence related data. The library was developed that can create client-side image of the popular web browsers. To overcome this problem, it would be most suited in this framework, because SVG use of XSLT stylesheet post-processing mechanism a report on the task's result is generated from the XML data using a description in an certain circumstances. At the end of a task run the objects will be stored on the server in an from this data. The methods of these objects can be used to give feedback into the work flow descriptions with regard to how to proceed in possible to create reusable transformation components For users of the task framework in a command-line mode, plain text versions of the task reports are also requested. If the transformation stylesheets are created carefully enough, the publicly available XSLT processors like Apache’s Xalan (XALAN, 2002) or Saxon (SAXON, 2002) transform the data for web users into HTML pages when called via W2H’s post-processor or into a plain text document using exactly the same stylesheet for text and HTML.

For a graphical reconditioning of the task’s result, the use of Scalable Vector Graphics (SVG; SVG, 2002) would be most suited in this framework, because SVG is just another application of XML and can be handled by XSLT. But SVG is currently not supported by some of the popular web browsers. To overcome this problem, a library was developed that can create client-side image maps in HTML from sequence related data. The library implements the public GD library (GDlib, 2002) to create the bitmap images needed in HTML. The definition of the map is done as part of the data flow description. The binary data is then stored as a raw output of the task flow and the annotation data becomes part of the main XML task output. By means of XSLT instructions the data is then merged again for the browser according to the HTML language.

**IMPLEMENTATION**

The task framework was built using the Perl programming language in order to take advantage in the language’s support for object-oriented programming with classes such as the framework’s job scheduling capabilities, the meta-data handling or the user interaction on the command-line and via W2H.

The application outputs are provided as objects to the author of a task. By querying the object’s methods, the task author can establish rules for the task’s work flow incorporating certain properties of application results. The parser engine was realized as a set of classes that can be inherited or used when certain application specific parser plug-ins (application parser classes) need to be added. Especially the possibility of class inheritance simplified the creation of several parsers, because it was possible to sub-class an abstract class, containing

---

**Fig. 1.** Sketch of the task framework: Using the work flow and data flow configurations for a task, W2H can request the user input and send it to the framework’s job scheduler. Its mission is to start and monitor the individual applications using the work flow description. The raw application outputs—by means of W2H’s output manager under user control—will then be passed to the parser engine that creates objects from this data. The methods of these objects can be used to give feedback into the work flow descriptions with regard to how to proceed in certain circumstances. At the end of a task run the objects will be stored on the server in an XML representation on disk. By means of W2H’s post-processing mechanism a report on the task’s result is generated from the XML data using a description in an XSLT stylesheet.
methods useful for a whole group of applications such as database searching tools. In this case for instance, methods containing code for parsing the lines of an HSP listing or an alignment block were overloaded, but accessor methods like next_alignment() or to_xml() were inherited from the parent. This means that the methods used by a task author remain the same, independently of the output type in use.

At the DKFZ, several work flows have been implemented within the HUSAR environment using the W3H task framework. PATH, the Phylogenetic Analysis Task in HUSAR (del Val et al., 2002), comprises several applications implementing different methods to estimate phylogenies.

PATH creates a report summarizing the consistencies and differences obtained, when comparing the results from phylogenetic inference using different methods like maximum likelihood, maximum parsimony and the distance method. Additional methods like the splits decomposition method are used in the work flow to complement the task report with information about the quality of the input data.

Other methods are used to assess the best set of parameters for certain parts of the analysis. After being provided with the input data by the web interface W2H, the PATH task description handles the preparation of the input data for the individual applications. Using the framework’s parser engine, PATH can evaluate the results of those applications to set the parameters for the following steps in the phylogenetic analysis accordingly and extract information for the final report. In Figure 2 a part of the task’s configuration is shown as an example.

Another task uses pattern and profile searches against several protein domain databases to annotate a given protein sequence with the hits obtained. The main part of the task description lies in the declaration of rules for the hit selection constraints, when evaluating the individual search results.

Another task processes unknown DNA sequences and shows several properties of this sequence together with putative functions. Additionally, tasks performing

---

Fig. 2. To illustrate the work and data flow configuration a part of the PATH configuration (Phylogenetic Analysis Task in HUSAR; del Val et al., 2002) is shown. This part describes the creation of two phylogenetic trees according to the distance and the parsimony method. RUN() statements allow the execution of external programs (in parallel). Variables with the $-prefix are used to connect the nodes of the data flow network. By declaring them as input or output, the job scheduler connects these variables with the data management of W2H.
secondary structure prediction and the classical EST clustering have been implemented using the W3H task framework.

DISCUSSION
The W3H task framework allows the integration of applications and methods already implemented within an existing software environment via the description of meta-data. Task designers with a background in life sciences will be enabled to create custom tailored analysis task flows (e.g. for high-throughput analysis). By specifying dependency rules between the used applications in the meta-data, tasks of high complexity can be designed. Through the framework’s connection with the web interface W2H, tasks are immediately available on the web without any additional web programming (using the meta-data).

First tasks covering the fields of phylogenetic analysis, protein domain databases, sequence property identification and secondary structure prediction have been implemented within this framework. Experience from this work and contact with other research groups using the HUSAR system has shown that there is a high demand for custom-tailored analysis tasks notably in the field of automated genome annotation.

Currently, the task’s meta-data still needs to be written in a Perl-like language. But we plan to facilitate the creation of work and data flow descriptions by employing an interactive task manager. With this manager the description language will be replaced by an XML-based meta-data language.

In order to use objects and parser plug-ins written in programming languages other than Perl, the language that was used to create the task framework, it will be necessary to add a CORBA (COPE, 2000; CORBA-EBI, 2002) or SOAP (SOAPlite, 2002) interface to the parser engine.

To support really interactive tasks where the task is configured to ask the user various questions during its execution, a mechanism to maintain the persistence of the task’s objects needs to be implemented, because of the stateless nature of the HTTP protocol used to communicate with web users. By an extension of W2H’s post-processing mechanism it is possible to achieve this user interaction via the web, transparent to the author of task flows.

ACKNOWLEDGEMENTS
We would like to thank the task designers Mechthilde Falkenhahn, Agnes Hotz-Wagenblatt, Barbara Pardon and Coral del Val for their valuable comments and Artjom Tarasenko for his participation in the development of the library for client-side HTML image maps.

REFERENCES